ピンポイント監視を可能にする柔軟なINT処理の実現

著者	佐藤 俊大
出版者	法政大学大学院情報科学研究科
雑誌名	法政大学大学院紀要．情報科学研究科編
巻	16
ページ	1-6
発行年	2021-03-24
URL	http://doi.org/10.15002/00023870

ピンポイント監視を可能にする柔軟な INT 処理の実現
A Flexible INT-based Pin-Point Network Monitoring

佐藤　俊大 *
Toshihiro Sato
法政大学大学院情報科学研究科情報科学専攻
E-mail: 19t0009@cis.k.hosei.ac.jp

Abstract

Telemetry is a new generation network monitoring technology, which is alternative of SNMP. Telemetry could reduce the load of requesting processes by getting information automatically on a regular basis. Among Telemetry, INT is receiving particular attention because this could escape the deterioration of measurement performance, by using existing packets to collect information. INT is embedding information and instructions for INT processes as INT header in packets. This enables INT compatible switches to work as instructions says, by reading INT header of received packets. But by simply implementing INT, all INT compatible switches will add information to packets with INT header. This causes explosive growth of amount of collecting information and makes the load of analysis much heavier. In this research, we realize INT processing platform which enables collecting information only with the switches required for analysis. Types of information and list of target switches are written in INT header, so switches could understand what kind of INT process is required by reading INT header. This enables reduction of overhead caused by embedding not needed information, and mitigation of complicated management and analysis of collected information.

1　導入

クラウド基盤上でのサービス運用や広域の通信インフラの管理の場面で，In-band Network Telemetry(INT) と呼ばれる技術が注目されている．INT では，通信の際に実際に流れているパケットに解析のための指示や情報を INT ヘッダとして埋め込む．これによって INT を用いた情報収集ではパケットの流量が変化しないため，Telemetry 技術の中でもパケットの流量が増加することによるネットワーク機器の性能低下を回避できることで注目されている．

INT が注目されるようになったのは，極めて大規模 (Hyper Scale) なネットワーク環境の登場によって SNMP などの従来のネットワーク監視技術では十分な監視が不可能になったことに起因する．SNMP は現代と比較して小規模かつ簡単な構成のネットワークを監視対象としており，必要に応じてネットワーク機器にリクエストを送信する形で監視を実現している．しかし Hyper Scale 環境が構築されるようになると，SNMP が Hyper Scale 環境に対して送信するリクエストの数は環境の規模に比例して膨大な量となる．そのため Hyper Scale 環境で SNMP を用いると，膨大な量のリクエストによってネットワーク機器の処理が停滞しネットワークのリアルタイム性が損なわれる．また，こうした中で Google などのネットワーク業界を牽引する組織は自社内の Hyper Scale 環境のリソースをクラウド基盤として他の企業や組織に提供するようになった．そのようなクラウド基盤では様々なマイクロサービスが多数連携して稼働しているためその実行結果の監視，特にサービス間の通信に関わる状況の監視や問題の解析のための測定が重要になる．しかし SNMP には上記のパフォーマンス面の問題以外にも，新しいデータモデルやプロトコルに対応していないという柔軟性の観点で大きな制約がある．そのため SNMP による監視では，サービス間の挙動や通信遅延などの状況の変動の詳細な分析ができず，クラウド上での複雑に連携したサービスを十分に監視することは難しい．

このほかに SNMP は定期的に情報を収集する際にネットワーク機器に大きな負荷がかかる問題もあり，監視の指示やデータ収集の処理負担が小さい INT が注目されるようになった．しかし INT は単純に実現すると，全ての INT 対応スイッチがパケットに対して INT 処理を実行する．INT では監視に用いる情報をパケットに埋め込むため，逆に INT に対応したスイッチは全て反応してしまい，監視対象ネットワーク内のスイッチの数や監視項目の数に比例してパケットに付与する情報が膨大なものになってしまう．この事態を回避するためには，INT 処理の対象となるスイッチや収集する項目をピンポイントに観測できる必要がある．そのため本研究ではスイッチ単体で INT 処理の実行の可否や収集する Telemetry 情報の種類を判断できるような INT 処理基盤を実現することで，不要な情報の収集を回避する．

2　関連研究

2.1　Sel-INT

Sel-INT[1] はコントローラベースの SDN によって INT 処理を実現し，スイッチ毎に INT データの項目やサンプリングレートをコントローラが柔軟に制御することで，必要以上の Telemetry 情報の収集を回避する．ここでは OpenvSwitch を機能拡張して，パケットを構成するビット列の特定部分に基づいて INT 処理ができるようにしている．OpenvSwitch のフローテーブルには予め INT ヘッダの有無，Telemetry 情報の種類，送信元がコントローラか否かに関するマッチ条件が記述されている．そしてコントローラは環境に合わせてサンプリングレートを決定し，サンプリングレートをスイッチに送信する．スイッチはコントローラから受信したサンプリングレートをフローテーブルに追記し，コントローラ以外から送信されてきたパケットにはフローテーブルに従い INT 処理を実行する．

* Supervisor: Prof. Toshio Hirotsu

2.2 PINT

PINT[2] は INT 情報を環境内の各スイッチが確率的に収集することで，全スイッチが INT 情報を収集することによって発生する負荷を軽減する．この研究は INT の収集時に発生するオーバーヘッドの削減のため，付与する情報に使うビット数を削減しつつも INT と遜色のない解析を可能にする．付与する情報を複数のブロックに分割し，個々のブロックを付与することで付与データ量を削減している．パケットとスイッチから計算されるハッシュ関数によって付与の有無と付与対象のブロックの指定を行うことで，全てのブロックが選ばれる確率を均一なものとしている．

2.3 P4

パケットを構成ビット列で直接制御する技術として P4[3] がある．P4 はパケットの制御を担当しており，パケットの入出力処理は BMv2 と呼ばれるソフトウェアスイッチが担当する．P4 は *Parse*，*Ingress*，*Egress*，*Deparse* の 4 つの段階に分かれてパケットを制御する．*Parse* ではパケットを分解する．一般的には Ethernet などのヘッダ別に分解するが，独自のヘッダ設計を記述することで任意の形に分解することが可能である．次の *Ingress* ではパケットを分解して得た各項目を，そのスイッチから出力される際にあるべき形に編集する．パケットの次の宛先 MAC アドレスを変更しスイッチのどのネットワークインターフェースからこのパケットを送出するかに関する情報を設定する．*Egress* では，パケットへの情報の追加を行う．このステップで行われる処理には，カプセル化のようにパケットに新たなヘッダを追加する処理が該当する．そして最後の *Deparse* にて，*Parse* で分解したデータをパケットの状態に戻す．このステップでは指定したデータのみがパケットの状態に戻るため，指定しないことでパケットから特定の要素を抽出することも可能である．

2.4 問題点

P4 などのプロトコル非依存なパケット制御技術を用い INT が必要とする INT ヘッダの構造や埋め込む位置を指定することで，パケットの指定箇所に指定の書式で埋め込まれた INT ヘッダを制御することが可能となる．しかしこれらの技術を用いて単純に INT を実装しようとすると，INT 対応スイッチが INT ヘッダを持つ全てのパケットに対して Telemetry 情報を埋め込んでしまう．これによって必要以上に Telemetry 情報を収集してしまい，パケットの転送時に不要な情報の分だけオーバーヘッドが増加することと分析の負荷が増大することが問題である．本研究では，分析のために必要となるスイッチのみで情報収集が可能となるような INT 処理基盤を実現する．各スイッチは INT ヘッダから付与する項目や収集地点に関する指示を読み込むことで実行するべき INT 処理を動的に判断できるようになる．コントローラは Telemetry 情報の種類と収集対象のスイッチのリストを指示として含んだ INT ヘッダをパケットに付与するだけで，スイッチや情報の種類を任意に選択することが可能となる．これによって不要な情報が付与することで発生する転送時のオーバーヘッドを削減し，データの管理・解析における煩雑さを軽減することが可能となる．

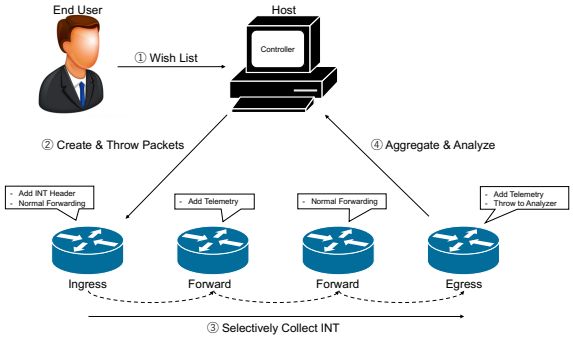

図 1　提案システムの構成

3 設計と実装

提案システムは指示を埋め込むコントローラと指示を読み込むスイッチ，そして Telemetry 情報を収集するデータ解析プラットフォームから構成される．コントローラはユーザが要求する INT 処理を実行するための指示を流れるパケットに挿入する．スイッチは流れてきたパケットの INT ヘッダから要求されている INT 処理を導出し，解析に必要となる Telemetry 情報の付与や分析のためのパケット収集を行う．必要な情報の付与が完了したパケットは直ちにデータ解析プラットフォームに収集され解析される．図 1 に提案システムの構成を示す．

3.1 INT ヘッダ

提案システムにおいてスイッチの INT 処理を制御するための INT ヘッダの構成を図 2 に示す．INT ヘッダは 6 種類の項目で構成される．*MapInfo* はこの INT ヘッダに付与される Telemetry 情報に対応したキー情報である．INT ヘッダには入力・出力ポート番号，ホップレイテンシ，帯域幅，スイッチの ID，タイムスタンプの 6 種類から，8Byte までのサイズで任意に組み合わせたものを Telemetry 情報として付与する．*Length* には INT ヘッダにこの Telemetry 情報がいくつ付与されているのかを表しており，*MapInfo* と組み合わせることで INT ヘッダのビットサイズを把握することが可能である．*etherType* にはパケットにおいて INT ヘッダの後に配置されているヘッダの種類が記述される．そして *Ingress* と *Point* を用いることで INT 処理を実行するスイッチの特定を実現しており，*Metadata* は INT ヘッダが保持する Telemetry 情報を指す．なお，図 2 における *Metadata* は Telemetry 情報のサイズが 1Byte の場合のデータの配置を示している．*Ingress* は，コントローラから数えて何ホップ目から INT 処理を開始するかを定めている．*Ingress* は INT ヘッダを含むパケットが INT 対応スイッチを通過するたびに 1 減少し，スイッチが INT ヘッダを解析した時点で *Ingress* が 0 だった場合のみスイッチは INT 処理を次のステップに進める．そして次の INT 処理のステップで，*Point* を用いてスイッチが INT 処理対象かどうかの判断を行う．*Point* において，各スイッチが処理の有無を判断するために参照するのはそのビット列の 1 桁目のビットである．この値が 1 だった場合に限り，スイッチは *MapInfo* で指定された Telemetry 情報を INT ヘッダに追加し，*Length* を 1 増加する．この時，*Metadata* は新しいものが *Point* の直

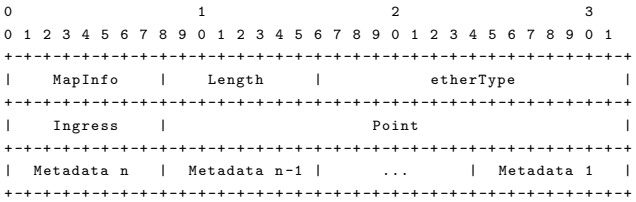

図 2　INT ヘッダの構成

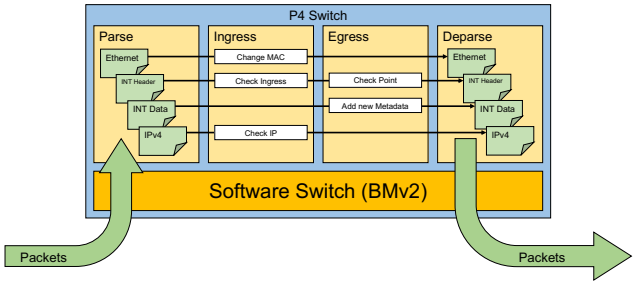

図 3　スイッチ内部の処理

後に来るように数珠繋ぎで挿入する．最後に INT 処理の有無の判断が終了したのち *Point* を 1 ビット右にシフトする．この処理によって 24 桁ビット列の *Point* に対応した，*Ingress* で指定されたスイッチから 24 ホップ間のスイッチにて INT 処理を制御することが可能になる．また *Point* が 0 になった場合，それはそのパケットにおける INT 処理が終了したことを意味する．そのため *Point* を 1 ビット右にシフトした結果 *Point* が 0 になった場合，そのスイッチはパケットから INT ヘッダを抽出し，それをデータ解析プラットフォームに転送する．

3.2　スイッチ

スイッチは P4 で実装し，INT ヘッダを読み込んで実行する INT 処理を判断する．図 3 にスイッチが INT 処理を判断する手順を示す．スイッチに届いたパケットはヘッダ毎に分解され，それぞれのデータ毎に適切な処理を行なったのちパケットの形に戻す．以下に P4 の各ステップでのパケット制御の詳細について述べる．

提案システムの *Parse* ステップでは，P4 はパケットの構造を Ethernet ヘッダ，INT ヘッダ，INT メタデータ，IPv4 ヘッダに分解する．P4 では同一のヘッダ構造が連続する場合は分解時に配列として一連のヘッダを保持する．提案システムにおいては複数個付与される INT メタデータがこれに該当し，*Parse* ステップでは INT ヘッダの *MapInfo* および *Length* から INT メタデータのサイズおよび個数を把握し，INT メタデータと同じサイズのヘッダを定義したのち，そのヘッダの配列を用意してパケットに付与されている全 INT メタデータをこの配列に格納する．なお，IPv4 ヘッダ以降の分解されない部分は 1 つの構造体として *Deparse* ステップまで保持される．

Ingress ステップではパケットを分解して得た情報を編集することで，パケットがスイッチから送出されたあとの動作を制御する．一般的にこのステップで変更される項目は Ethernet ヘッダの宛先 MAC アドレスや P4 が保持するパラメータのうち送出ポートを担う変数など，次にパケットが向かうスイッチでパケットが正しく処理されるためのものである．提案システムではこれらの値の編集に加えて，INT ヘッダの *Ingress* の参照および減算処理を行う．

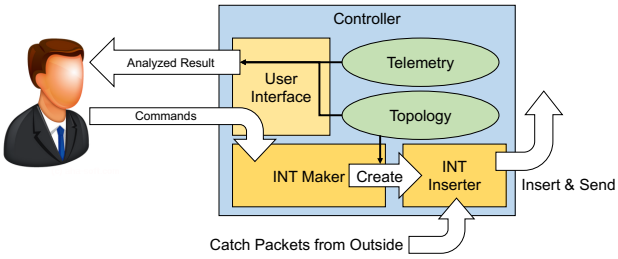

図 4　コントローラの処理

Egress ステップではパケットに新たな Telemetry 情報を追加する．P4 には配列の先頭に空の要素を追加する *push_top* という組み込み関数が存在するため，*Point* の 1 桁目から INT 処理の実行が決定した場合はこの関数を実行することで INT メタデータ配列の各要素を 1 つずつ後ろにずらし，先頭の要素を空にする．なお，P4 においてこのように追加された要素は初期設定では無効なヘッダ構造として認識されているため，組み込み関数の *setValid* を用いてヘッダを有効にする必要がある．メタデータ配列の先頭を有効にしたのち，*MapInfo* に従い Telemetry 情報を INT メタデータ配列の先頭に書き込み，*Length* の値を 1 増加する．

最後の *Deparse* ステップで，P4 が *Parse* ステップで分解し *Ingress*，*Egress* ステップを経て編集されたデータをつなぎ合わせてパケットの形に戻す．*Parse* ステップで分解した時と同様に，Ethernet ヘッダ，INT ヘッダ，INT メタデータ，IPv4 ヘッダの順番で宣言することでパケットの本来の構成に戻すことが可能である．また，INT ヘッダの抽出などパケットに戻さなくて良い情報がある場合は，このステップで戻すよう宣言しないことでパケットから取り除くことが可能である．なお，*Parse* ステップで宣言せず分解されなかった部分は明示的に順番を指定した項目を並べ終えた時点で末尾の項目の後ろに自動的に配置される．

3.3　コントローラ

コントローラはネットワーク環境において INT 処理可能な範囲の境界部分にエッジノードとして配置する．ユーザはこのエッジからコントローラにアクセスし，INT 処理を実行するスイッチと収集する Telemetry 情報を指定する．コントローラはユーザからの指示を受け，要求通りの処理を実現するような INT ヘッダを通過するパケットに付与する．このような動作をするコントローラをエッジに配置することで，環境外部から内部に向けて転送されるパケット全てに INT ヘッダを埋め込み，ユーザが要求する INT 処理を実現することができる．

図 4 にコントローラの処理の全体像を示す．コントローラはユーザインターフェース，INT ヘッダ設計機能，INT ヘッダ挿入機能の 3 つの機能から構成され，Telemetry と Topology の 2 つのデータベースを保持する．Telemetry はネットワーク環境内で収集した情報のうち最新の情報のみを集めたデータベースであり，Topology は監視するネットワーク環境を構成するスイッチのトポロジーをまとめたデータベースである．

ユーザインターフェースは監視情報の表示機能とユーザからの INT 処理に関する指示を取得する機能から構成される．表示機能は監視情報を画像化したものを表示する．表示機能は最

初に Topology データベースから環境内の全スイッチの ID およびそれらのスイッチ間のリンクを取得する．次に Telemetry データベースから各スイッチから取得した最新の Telemetry 情報を取得し，各 Telemetry 情報をスイッチの ID に紐付ける．そして全てのスイッチの ID を記したアイコンを表示し，リンクを元にそれらのアイコンを線で結ぶ．その後に Telemetry 情報の数値に応じてアイコンの色分けを行うことで，情報を収集しているスイッチやその中でも異常な数値を示しているスイッチを把握しやすいマップ画像を生成している．そして指示取得機能はコントローラ内で常駐しており，ユーザは監視中にこの機能を用いて任意のタイミングで収集する Telemetry 情報や収集対象のスイッチを変更することが可能である．

INT ヘッダ設計機能はユーザインターフェースの指示取得機能によって呼び出される．ユーザインターフェースを介して取得したユーザからの INT 処理の指示と Topology データベースから取得したネットワーク環境を元に，ユーザの指示に沿った INT 処理を実現するために必要な情報の種類やパケットが通過するべき経路を決定し，それらの設定を INT ヘッダ挿入機能に伝える．

INT ヘッダ挿入機能はコントローラ内に常駐しており，INT ヘッダ設計機能からの連絡とコントローラに届くネットワーク環境内外からのパケットを監視する．ここで INT ヘッダ挿入機能が設計した INT ヘッダの構造を保持し，ネットワーク内部に向かう全てのパケットにその INT ヘッダを挿入することでユーザからの指示に沿った INT 処理を環境内のスイッチで実現する．また，ネットワーク環境外に出るパケットに INT ヘッダが挿入されていた場合は環境外におけるパケットの処理に弊害が生じるため，環境外に向かうパケットからは INT ヘッダを抽出する．

3.4 データ解析プラットフォーム

データ解析プラットフォームは，特定のネットワークインターフェースから受信したパケットから Telemetry 情報を抽出して保管する．図 5 にデータ解析プラットフォームの処理を示す．データ解析プラットフォームは Moloch と解析プログラムの 2 つの機能から構成され，Elasticsearch と TelemetryDB の 2 つのデータベースを有している．

Moloch は MolochViewer と MolochCapture の 2 つの機能を有するアプリケーションである．MolochCapture は監視対象のネットワークインターフェースを指定することでそれらのネットワークインターフェースに届いたパケットを収集する監視タスクである．MolochCapture が収集したパケットは Elasticsearch データベースに格納されていく．MolochViewer は Elasticsearch データベースに格納されたデータを閲覧するための GUI ベースのインターフェースを提供する．Moloch を用いることで Telemetry 情報の収集を終えたパケットが回収され，パケットの生データが Elasticsearch に蓄積される．

解析プログラムは Python で実装しており，Moloch が対応していない INT ヘッダの解析を担う．Moloch は API を用いることで Moloch のシステム関連のデータをリスト形式で取得することが可能である．解析プログラムはこの Moloch API を用いて Elasticsearch に保管された全パケットデータのファイルパスを取得し，取得したパケットの生データを解析する

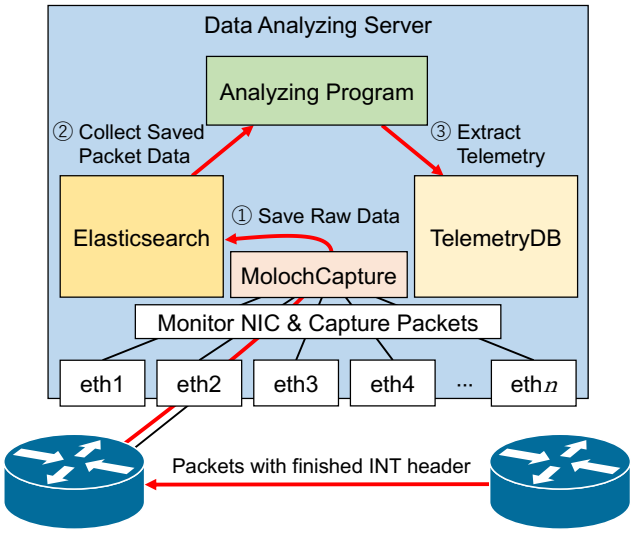

図 5　データ解析プラットフォームの処理

ことで INT ヘッダに格納された Telemetry 情報を解析する．また，解析プログラムはコントローラが保持するユーザからの入力内容を取得することで，監視範囲と収集項目を認識する．これは，INT ヘッダの処理上の問題で Telemetry 情報の収集が完了した時点で INT ヘッダの *Ingress* および *Point* が 0 になっており，INT ヘッダからでは Telemetry 情報の数やパケットが通過した経路は判明しても Telemetry 情報を付与したスイッチを特定することができないためである．このようにコントローラと同様の処理によってユーザの入力内容から *Ingress* と *Point* の値を導出することで，解析プログラムは INT ヘッダに付与された Telemetry 情報を付与したスイッチを特定し，各 Telemetry 情報とスイッチを紐付け他解析が可能となる．

4 実験と評価

提案システムが INT 処理を実行するスイッチを状況に応じて柔軟に選択できることを示すため，コントローラからスイッチを相対位置・絶対位置で指定する実験を行った．また，提案システムが INT 処理を実行するスイッチを選択することの有用性を示すため，パケットに付与される INT 情報の量がパケット転送時のオーバーヘッドに与える影響を調査した．実験は Topology Zoo が公開しているトポロジーデータの LambdaNet を Mininet で再現した環境で実行した．

4.1 トポロジー解析

1 つ目の実験として，提案システムがコントローラからスイッチを相対位置で指定できることを示すためにトポロジー解析を行った．Mininet 環境にてコントローラとして配置されたノードはスイッチのトポロジーを認識していない状態であり，そこから Telemetry 情報のみでトポロジー解析を行う．

コントローラはパケットに対して以下 2 点の INT 処理を実現するための INT ヘッダをパケットに挿入する．

1. 取得する Telemetry 情報は隣接するスイッチ 2 台の ID
2. INT 処理ののち，スイッチはそのスイッチが隣接する全てのスイッチにパケットを送出

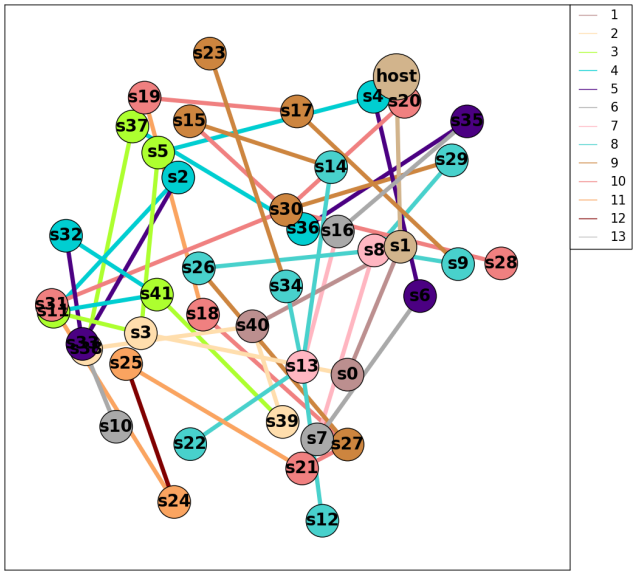

図 6　生成されたトポロジー解析マップ

その上でコントローラは ID を取得するスイッチのホップ数を順に上げていき，解析パケットを繰り返し送出することでネットワーク環境内の全てのスイッチの ID を取得する．データ解析プラットフォームは収集したパケットから INT ヘッダを抽出し，そこから隣接するスイッチの ID のペアをリンク情報として取得する．データ解析プラットフォームは収集したパケットから未知のリンクが取得できなかった段階で環境を構成するスイッチ間のリンクを全て取得できたと判断し，保存されたパケットの解析を終了する．

図 6 はコントローラが最終的に生成したトポロジーマップを画像化したものである．データ解析プラットフォームが収集するスイッチ ID のペアはホストから徐々にホップ数を増加させている．そのためコントローラはこのリンク情報をデータ解析プラットフォームから取得することで，直近のスイッチを中心として認識しているネットワークトポロジーを放射状に拡張していく．図 6 はリンクをそれぞれ何回目の拡張の時点で発見したかによって色分けしており，13 回目の拡張によってトポロジーの解析が終了したことを示している．トポロジーマップを完成させられたことで，提案システムではホストからのホップ数によるスイッチの相対的な指定が可能であることを示した．

4.2　監視中の柔軟な解析内容の変更

2 つ目の実験として，コントローラがスイッチを絶対位置で指定できることを示すために，ネットワーク監視中に状況に応じた INT 処理内容の変更が可能かを調査した．この実験では絶対位置の指標としてスイッチ ID を用いる．これは，スイッチ ID によって指定されるスイッチはコントローラがネットワークのどのエッジに配置されるかに関わらず一つに定まるためである．

監視の最中に特定のスイッチのみを監視するよう INT 処理内容を変更した場合に，その設定が迅速に反映されるかどうかをユーザインターフェースに表示される監視マップの内容から判断する．監視の初期段階ではコントローラから複数のホストまでの経路にある全てのスイッチのホップレイテンシを収集する．ホップレイテンシはスイッチにパケットが到達してから送出されるまでに経過した時間であり，ミリ秒単位で収集される．こうして複数のスイッチのホップレイテンシ情報を反映させた監視マップが図 7 である．図 7 から，複数のスイッチのアイコンが緑や赤に着色されていることが分かる．各スイッチのアイコンは取得したホップレイテンシの大きさを示しており，緑に近いほど小さく，赤に近いほど大きく，そして白は収集対象外であることを示している．図 7 でホップレイテンシが大きいことが示されている S8 のみを監視するように設定を更新した結果表示された監視マップが図 8 である．図 8 から S8 以外

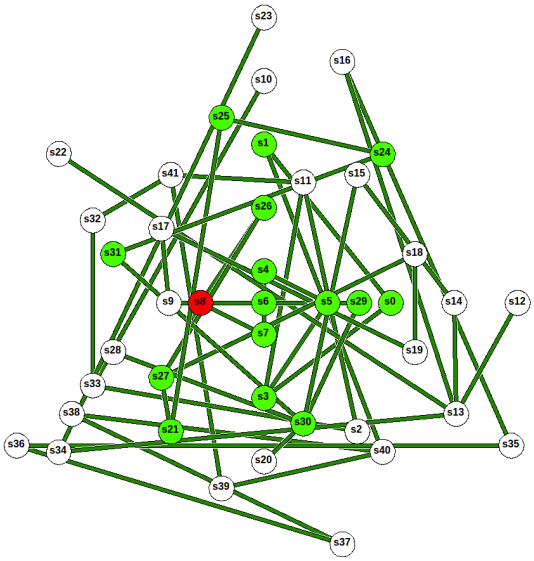

図 7　複数のスイッチの状態が表示されている監視マップ

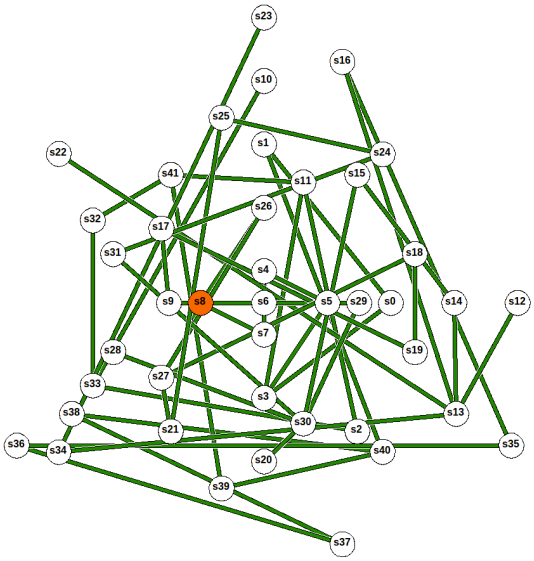

図 8　S8 以外の Telemetry 情報を収集しなくなった監視マップ

表 1　全体 (Full)・部分 (Part) 収集時のパケット転送処理のホップレイテンシ (ミリ秒)

	s1	s40	s38	s37	s36	s35	s16	s13	s34	s23
Full	20	11	10	14	23	54	41	62	65	71
Part	17	-	-	9	-	-	10	-	-	35

のスイッチのアイコンが白いため S8 以外から Telemetry 情報は収集されていないことが分かる．このことから，提案システムではスイッチ ID によるスイッチの絶対的な指定が可能であることを示した．

4.3　特定のスイッチのみからの Telemetry 収集

3 つ目の実験として，提案システムが INT 処理対象スイッチを指定できることの有用性を示すため，INT 処理対象スイッチの数がパケット転送処理時間に与える影響を調査した．この実験では環境内の 10 個のスイッチを通過する経路を用い，経路上の全てのスイッチで INT 処理を実行した場合と特定のスイッチでのみ INT 処理を実行した場合に収集したホップレイテンシを比較する．

表 1 は 10 個から収集した場合と 2 個飛ばしに収集した場合のホップレイテンシを 100 回測定した平均値を示す．表 1 から収集対象のスイッチを限定することでホップレイテンシが減少し，スイッチ内でより短い時間でパケットの処理を完了したことが分かる．このことから，INT 処理を実行するスイッチを限定することでパケット転送時に発生するオーバーヘッドを削減できたことが分かる．

5　考察

トポロジー解析実験の結果から，提案システムはコントローラからのホップ数という相対位置に基づいてネットワーク内のスイッチを指定することができることを示した．また，ホップ数を段階的に増加してスイッチ ID を取得していくことでネットワーク内部のスイッチのトポロジーデータを生成できることが判明した．監視中に INT 処理内容を変更する実験の結果から，提案システムではコントローラでスイッチ ID から収集対象となるスイッチを指定できることを示した．このことから，提案システムにおいてはスイッチの ID さえ判明していればコントローラとスイッチ間の連携がなくとも柔軟に監視対象スイッチを変更できることが判明した．そして提案システムにおいてスイッチの ID はトポロジー解析によって容易に取得できることから，提案システムは一切のトポロジーが不明なネットワークにおいてもスイッチや Telemetery 情報をピンポイントに指定して収集することが可能である．従来の INT 監視技術はコントローラとスイッチの連携が必要であったり任意の情報の選択が不可能であることから，提案システムは従来の INT 監視技術と比較して高い汎用性を有していることを示した．

また Telemetry 収集対象のスイッチを限定してパケット転送時のスイッチの処理時間を測定した結果，経路上の全てのスイッチから Telemetry 情報を収集するよりも短い時間でスイッチのパケット転送処理が終わることが判明した．このことから，提案システムを用いてスイッチレベルで選択的な INT 処理を実行することは，パケット転送時のオーバーヘッドを削減する上で有効であることが分かる．

しかし実験を通して，提案システムに課題と疑問が判明した．課題は収集可能な範囲である．実験において用いた LambdaNet トポロジーは小規模なネットワークだったため提案システムの INT ヘッダで環境内を網羅的に監視できたが，ネットワーク環境の規模によっては提案システムでは網羅的な監視ができない可能性がある．その原因は *Ingress* と *Point* の仕組みにある．*Ingress* の最大値は 255 であるため，提案システムではコントローラから 255 ホップまでしか INT 処理をスキップさせることはできない．また，*Point* は桁数に合わせて最大で 24 個の連続するスイッチでしか INT 処理の実行の可否を指定できない．そのため，提案システムにはコントローラから *Ingress* と *Point* の合計である 279 ホップ離れたスイッチまでしか指定できないという制限がある．この制限のため，コントローラから最も遠いスイッチまでのホップ数が 279 を超えるネットワーク環境においては十分な監視を行えない．

総評として，提案システムによるスイッチレベルで選択的な INT 処理の実現は限定的ながら可能である．提案システムはスイッチの相対的・絶対的な指定が可能でトポロジー不明なネットワークにも用いることができるため汎用性が高いこと，そして Telemetry 情報の収集地点を選ぶことでパケット転送時のオーバーヘッドを削減できることを示すことができた．

6　結論

本研究では，スイッチのみで処理内容を判断できる柔軟な INT 処理基盤を実現した．評価実験からコントローラは詳細不明なネットワーク環境においてもピンポイントにスイッチや項目を指定しての INT 処理が可能であることを示した．また，提案システムを用いて情報の取捨選択を行うことはパケット転送時のオーバーヘッドを削減できる点で有用である．

しかし同時に，提案システムによる監視は限定的なものであり，監視対象環境の規模や複雑さによっては満足な監視が実現できないことも判明した．今後としては，提案システムへのよりホップ数の離れたスイッチでの INT 処理を実現する．

文　献

[1] S. Tang, D. Li, B. Niu, J. Peng, and Z. Zhu. Sel-INT: A Runtime-Programmable Selective In-Band Network Telemetry System. *IEEE Transactions on Network and Service Management*, Vol. 17, No. 2, pp. 708–721, 2020.

[2] Ran Ben Basat, Sivaramakrishnan Ramanathan, Yuliang Li, Gianni Antichi, Minian Yu, and Michael Mitzenmacher. PINT: Probabilistic In-Band Network Telemetry. In *Proceedings of SIGCOMM '20*, p. 662–680, 2020.

[3] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.*, Vol. 44, No. 3, p. 87–95, July 2014.